RESEARCH PLAN PROPOSAL

Performance Enhancement Techniques of Cloud Database Queries

For registration to
Doctor of Philosophy

IN THE FACULTY OF COMPUTER SCIENCE

to

THE IIS UNIVERSITY, JAIPUR

Submitted By:

Ruchi Nanda

Under the Supervision of:

Prof.(Dr.) K.S.Sharma
Advisor,
The IIS University

Prof. (Dr.)Swati V. Chande
Principal Computer Sc.,
International School of
Informatics & Management

Department of CS & IT

April, 2011

Performance Enhancement of Cloud Database Queries

1. Introduction

Cloud Computing

In an attempt to gain a competitive edge, enterprises are continuously trying to reduce their computing, administrative and infrastructure cost. This has forced the realization of new innovative technologies. Cloud computing is one of the technologies that have received major attention.

Marks et al. (2010) defined Cloud computing as a type of computing that provides simple, on-demand access to pools of highly elastic computing resources. These resources are provided as a service over a network (Internet). Cloud enables the customers of the technology to think of computing, effectively limitless, at minimal cost, and with high reliability, as well as not being concerned about how it is constructed, how it works, who operates it, or where it is located.

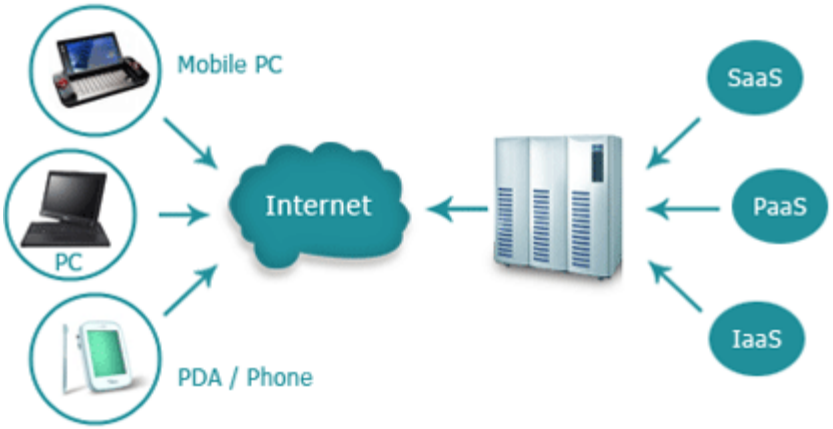


Fig.1. Cloud Computing [24]

In other words, cloud is a network of servers that pools different resources and cloud computing is a computing where the customers can access those servers using a web browser regardless of their device and location. A cloud provides on demand access to the resources and automatically deprovisions them when there is no demand.

As per Hurwitz et al. (2010), Cloud computing has three main tasks, that can be automated and delivered to customers in a consistent manner. These tasks are referred to as services which are as follows-

1. Infrastructure as a Service (IaaS): it offers storage and computing resources as a service
2. Platform as a Service (PaaS): it offers development and deployment environment for customers applications
3. Software as a Service (SaaS): it offers purpose-built business applications.

Cloud computing has four deployment models (Hurwitz et al (2010), Marks et al (2010)):

1. Public: The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.
2. Private: The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.
3. Community: The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy,

and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.

4. Hybrid: The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting).

Characteristics of Cloud computing (Marks et al., 2010) are as follows:

1. Elasticity and Scalability: the cloud is elastic, i.e. the resources can be provisioned or de-provisioned on-demand. Elasticity enables scalability, which means that the cloud can scale upward for peak demand and downward for higher demand.
2. Self-service provisioning: Customers can request for an amount of computing, storage, software, process or other resources from the service provider. After they use these resources they can be automatically de-provisioned.
3. Standardized interfaces: Cloud services should have standard application programming interfaces (APIs), which provide instructions on how two applications or data sources communicate with each other. A standardized interface lets the customer more easily link cloud services together.
4. Billing and Service usage metering: Customers can be billed for resources they use. They have to pay only for the resources they use. This is termed as "pay-as-you-go" model.

Data Base Management Systems

Jain (1997) defined data as the basic facts about the activities of the system, may be a business house, a production center or an educational institution. Data is organized into a more useful form, which is referred to as information. It helps organizations in their decision making process. Ramakrishnan et al. (2000) define database as a collection of data, typically describing the activities of one or more related organizations.

An efficient database management system (DBMS) is necessary for the creation, organization, management and retrieval of data from the database. DBMS consists of a variety of database models such as the network model, hierarchical model and relational model.

The most widely used data model is Relational data model. It was first introduced by Edgar Codd in 1970, A Relational model is the basis for any relational database management system (RDBMS). It uses a collection of tables to represent data items and their relationships. It is well-suited for Online Transaction Processing Systems, which are more heavily loaded with interactive transactions. These applications typically rely on the Atomicity, Consistency, Isolation, and Durability (ACID) guarantees that databases provide. These traditional DBMS are also known as row-oriented DBMS, as they store the data row-by-row. They keep all the information about an entity together and are preferable when workload tends to access data on the granularity of an entity. Some of these row-oriented DBMS (Fig.2) includes Oracle, DB2, SQLServer, Teradata.

The performance of any database management system can be determined by the following factors:

- System level issues like Hardware, CPU, I/O, memory.
- Database design which includes the logical and physical database design schema like table, constraints, and file organization
- Query Processing and Optimization techniques

As per Ramakrishnan et al. (2000), the ease with which information can be obtained from a database often determines its value to a user. As per Connolly (1988), a significant amount of

research has been devoted to developing highly efficient algorithms for processing queries. There are many ways in which a complex query can be performed, and one of the aims of query processing is to determine which one is the most cost effective.

The function of query processing (Fig.3) is to transform the query written in a high-level language into a correct and efficient execution plan expressed in a low- level language. An important aspect of query processing is query optimization. As there are many equivalent transformations of the same high-level query, the aim of query optimization is to choose an efficient execution plan for processing a query. It chooses the one that minimizes the resource usage by using the information from the system catalog. The set of query plans selected for examination is formed by examining the possible access paths (e.g., index scan, sequential scan) and join algorithms.

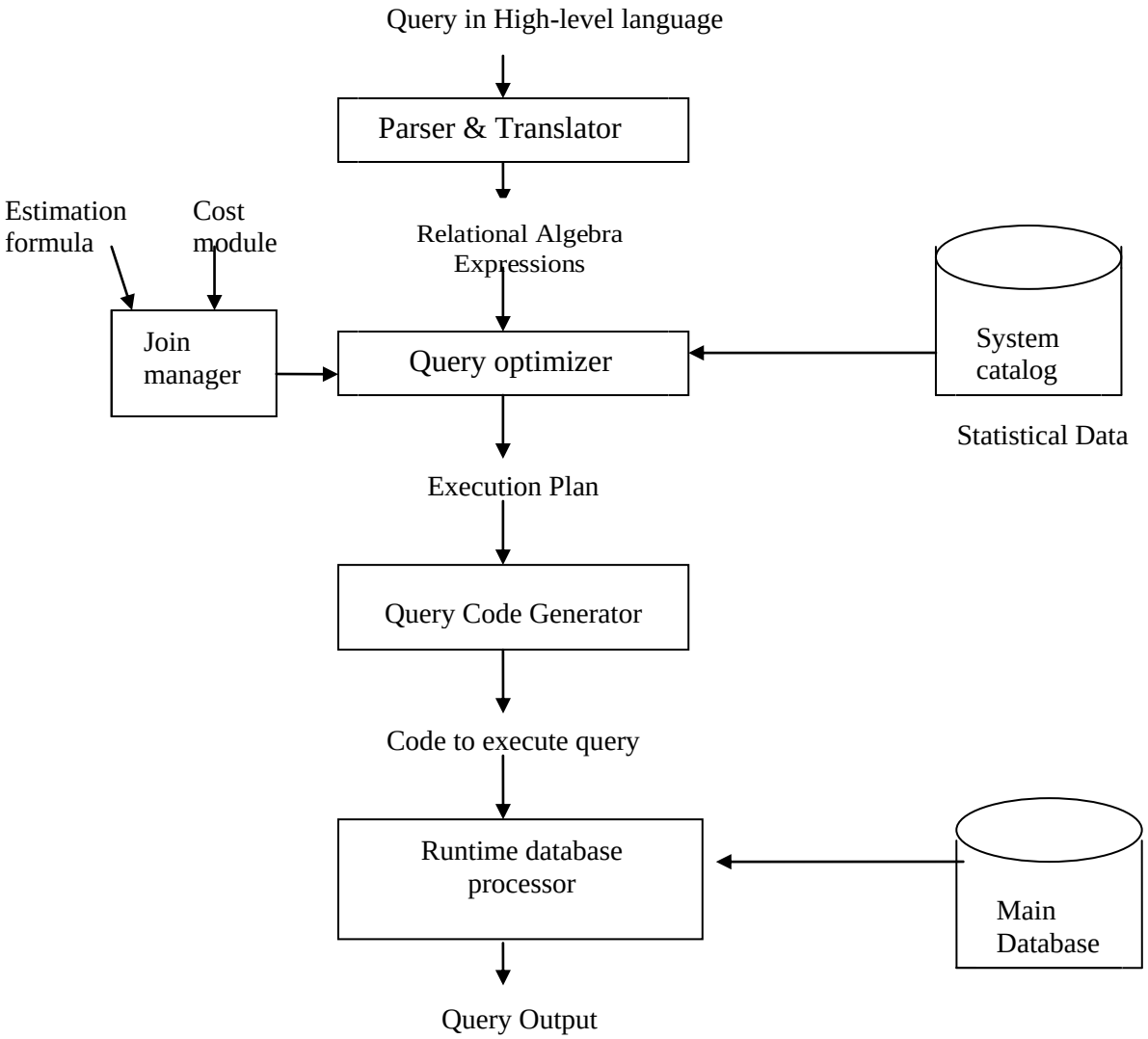


Fig 2. Steps in Query Processing

As the volume of the data is increasing since the last fifteen years at an exponential speed, research is continually needed to improve the performance and speed of data retrieval from the database management system. There is need for such database management system, which can process highly complex queries and handle terabytes or petabytes of data. Analytical applications are in demand, so that proper planning and decisions can be made from the data stored in the database.

Databases in Cloud Computing Environment

In cloud computing environment, data is distributed across a vast number of servers and costumers have little control on them, so an effective management of data is a big concern for organizations using cloud-based services. Here, the most important requirement of database management system is of scalability. Though RDBMS are robust and a vast majority of current database systems are based on the relational model yet they are not well-suited to scaling across large clusters of servers as they are not designed to be distributed. So it is difficult to ensure consistency, referential integrity and query performance in a distributed relational environment. Structured Query Language (SQL) is the dominant language of RDBMS. Wiggins (2000) stated that SQL databases don't scale.

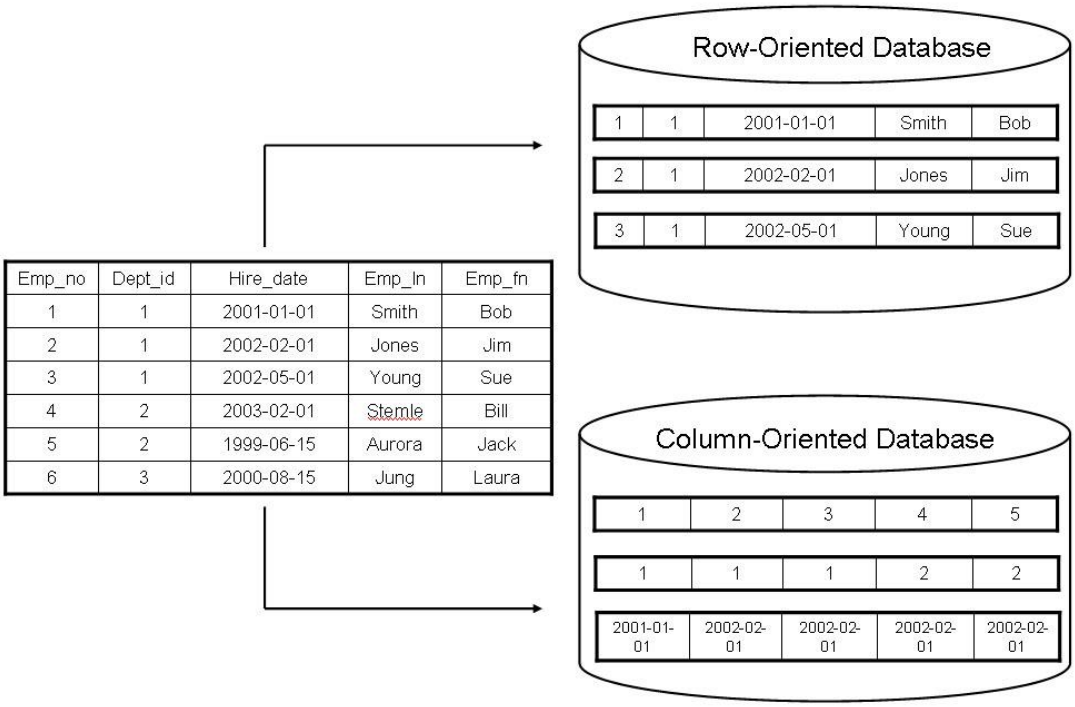

Row-Oriented Database
1 1 2001-01-01 Smith Bob
2 1 2002-02-01 Jones Jim
3 1 2002-05-01 Young Sue
Emp_no Dept_id Hire_date Emp_ln Emp_fn
1 1 2001-01-01 Smith Bob
2 1 2002-02-01 Jones Jim
3 1 2002-05-01 Young Sue
4 2 2003-02-01 Stemle Bill
5 2 1999-06-15 Aurora Jack
6 3 2000-08-15 Jung Laura
Column-Oriented Database
1 2 3 4 5
1 1 1 2 2
2001-01-01 2002-02-01 2002-02-01 2002-02-01 2002-02-01

Fig 3. Row-oriented and Column-oriented Databases [12]

Hence alternatives to relational database management systems have been developed. They are termed as "NoSQL datastores" and considered an alternative to SQL. They are also considered as "Not only SQL". They usually neither have a fixed table schema nor use the joins. They employ a distributed architecture and data is kept on several servers in a redundant manner. Some of the alternatives to RDBMS are:

1. Column-oriented DBMS (Fig.2) is a database management system that stores its content by column rather than row. It is faster to read and allows a more efficient compression of the data. Hence query processing time is decreased. These are well-suited for online analytical processing systems (OLAP) and data warehouses. Some of the database management systems which are already in the market are Vertica (a commercial version of C-store), SADAS as well as open-source DBMS like LucidDB and MonetDB.

2. Key-value Stores (non-relational DBMS) are used for storing large scale data & provide easy access. Its data models are schema-free, join-less and horizontally scaled. Here, domain is just like a table, but there is not a predefined schema. It is like a bucket where we can put the items. Items are identified by keys and a given key can have dynamic set of attributes attached to it. Data is created, updated, deleted and retrieved using API method calls. Some of its projects include Dynamo used by Amazon.com, Google's Bigtable used in the Google's application, Cassandra used by Facebook for inbox search and project Voldemort used by LinkedIn.
3. Mapreduce (Dean & Ghemawat, 2004) is a programming model to support distributed computing of large sets of data.

As discussed above, the performance of any database depends upon hardware resources, database design, query processing and optimization techniques. Generally DBMS automatically manages hardware resources. So it reduces the need for extensive manual testing. In cloud environment, as the data is distributed and scalability is the major concern, so query is processed and optimized in a different way.

The research in the cloud computing field is in its initial stage and it has the potential for advance research/discoveries by making data and computing resources instantaneously available. Moreover, the cloud databases are the future of enterprises' databases. Hence, there is enough scope in the database domain of cloud computing environment. The studies carried out so far indicate that the efficient data management is needed to handle large volumes of data and to support concurrent end-users.

2. Review of Literature

Wu et al. (2010) explained that the query efficiency is achieved by employing a pure key-value data model where, both key and value are arbitrary byte strings (e.g. Dynamo), or its variant (as in Bigtable), where key is an arbitrary byte string and value is a structured record consisting of a number of named columns. The authors pointed that these solutions lack support for secondary indexes, range queries and multidimensional queries.

Dean et.al (2004) proposed a programming model and a framework "Mapreduce" for processing large sets of raw data. A map-reduce program consists of two functions: Map and Reduce. The Map function processes the input data by distributing them to worker nodes for parallel computation and produces a set of intermediate results as key-value pairs, while the reduce function aggregates all the intermediate results with the same key from each node to produce the result. It can be used for structured data analysis of large sets. The limitations of Mapreduce as given by (Wu et.al (2010)) are:

- It produces the necessary secondary indices in an offline batch manner. Hence, secondary indexes are not up-to-date. So newly inserted rows cannot be quired until they are indexed.
- It does not provide data schema support, declarative query language and cost-based query optimizations.

Wu et.al (2010) proposed Cloud Global Index (CG-Index), which is a secondary B^+-tree based indexing scheme for cloud storage systems. It provides high scalability, high throughput, high availability and high concurrency. It provides range search and dictionary operations. The authors used generic key-pointer representation, partition aware indexing and eventual consistency

techniques. Index distribution technique for desired scalability was used. Experiments on Amazon's EC2 were carried on and the results demonstrated that it handles a mixed workload of queries and updates efficiently, but the main limitation of CG-index was that it supports one-dimensional queries only.

Stonebarker et.al. (2005) proposed C-store which is a column oriented store. Abadi et.al. (2006) demonstrated that the performance of the query processing can be increased if operations are directly applied on the compressed data. By employing this point of view they extended the C-store further. They applied algorithms commonly used by traditional DBMS and also applied algorithms especially suited for column-oriented systems on C-store. Finally comparison was made by changing the parameters like query workload and size of the data set. The results showed that the performance benefits of operating directly on compressed data in column oriented schemes is much greater than the benefits in operating directly on row-oriented schemes. They created decision-tree to aid the database designer to decide how to compress a particular column. The limitation with the optimizer was that it is not aware of the decompression costs of the various compression algorithms.

To optimize the execution of queries a number of greedy and approximation algorithms have been proposed earlier. But Kalnis et. al. (2003) stated that they do not scale well for realistic workloads. They developed two greedy algorithms which emphasize on finding the most beneficial view in each step instead of finding most promising query. Their extensive experiments showed that their methods outperform the existing one.

Kossmann & Stocker (2000) proposed a new class of query optimization algorithms, known as Iterative dynamic programming. The authors worked on common sub-expressions to reduce evaluation cost. It produces the best plan of all known algorithms in which dynamic programming is not viable because of its complexity. Though dynamic programming is viable, some IDP variants are adaptive and produce better plans. All the IDP algorithms can be very easily integrated into an existing optimizer, which is based on dynamic programming.

Manegold et.al (2000) presented an efficient inter-application multi-query optimizer that re-uses previously computed (intermediate) results and eliminates redundant work. Experimental results on a single CPU system and a parallel system showed that the inter-application multi-query optimizer improves the query evaluation performance significantly. The authors organized the optimization in 3 tiers: strategic, tactical and operational optimization. At each tier, different sources of optimization were exploited. The optimizer reused previously computed results and eliminated common subexpressions. It chooses at run-time either hash-join or nested-loop join algorithms. Performance experiments with Drill Down benchmark showed that the optimizer yields significant improvements of upto a factor of 4. The limitation of inter-application optimizer is that the optimizer is limited to identify and re-use equivalent intermediate results, only. It is also beneficial to re-use the smallest superset of a requested intermediate result in case the equivalent result is not available, provided that using the superset is cheaper

The studies show that there is enough scope in the performance improvement of queries as there are certain limitations with the works reported in the past and the same have a scope for further improvement.

3. Motivation

With the increasing volume of data across the large number of applications, the challenge is to distribute the computations, responding to the query along with the distribution of data. As per Bobrowski (2011), relational database management systems have grown overly complex, difficult to manage, and are struggling today to take full advantage of cloud computing technology. So, there is need to reanalyze the design and processing of relational database technologies and refine the existing methods or develop new approaches exclusively for the cloud environment.

Query processing and optimization are very important and necessary functions for any data base management system. The efficiency of query optimization algorithm is crucial to the performance of a DBMS. Hence query optimization has been a subject of research for many years because of the needs of the modern database applications. There are issues like query structure, access methods and join-order which need consideration in query optimization.

As the existing algorithms lack to meet the requirement of scalability and on-demand access of cloud databases, so there is need to improvise the same or develop new algorithms. As such, there is an enormous scope in the area of query optimization algorithms since cloud computing is an emerging field.

4. Objectives

The objectives of the proposed research are:

1. To explore the architectural and operational characteristics of cloud databases.
2. To analyze the factors affecting the performance of a query processor in cloud databases.
3. To analyze the existing techniques of processing and optimization of queries in the cloud-based databases.
4. To suggest necessary improvements in the existing techniques of processing and optimization of queries from cloud-based databases for improved efficiency.
5. To suggest a viable alternative to existing techniques for the processing and optimization of queries.
6. To improvise the existing algorithms or design new algorithm(s) for optimal utilization of resources for fast and accurate retrieval of data from the cloud-based databases.

5. Methodology

To systematically solve the research problem and to achieve the objectives, we propose to use the following methodology:

1. Study the architectural characteristics of cloud databases using secondary sources.
   For this, we will review the differences in the characteristics of cloud databases and as performance is our major concern, we propose to analyze the critical issues of query processor(s) in the cloud databases.
2. Analyze the factors affecting the performance of query processor in cloud databases.
   For this, we will perform a comparative analysis of the differences and gauge their criticality.
3. Analysis of the performance of cloud databases on the basis of their query processing and optimization techniques.

After having identified the critical factors we plan to study the existing algorithms for query processing and optimization in cloud databases.

4. Suggest necessary improvement in the existing techniques of processing and optimization.

   This involves finding the areas of improvement and suggesting improvements in the existing algorithms of the query optimizer for improved efficiency.

5. Suggest a viable alternative to existing techniques for the processing and optimization of queries.

   On the basis of the improvement in the existing algorithms, new algorithms or techniques for fast and accurate retrieval of data from the cloud databases can be proposed.

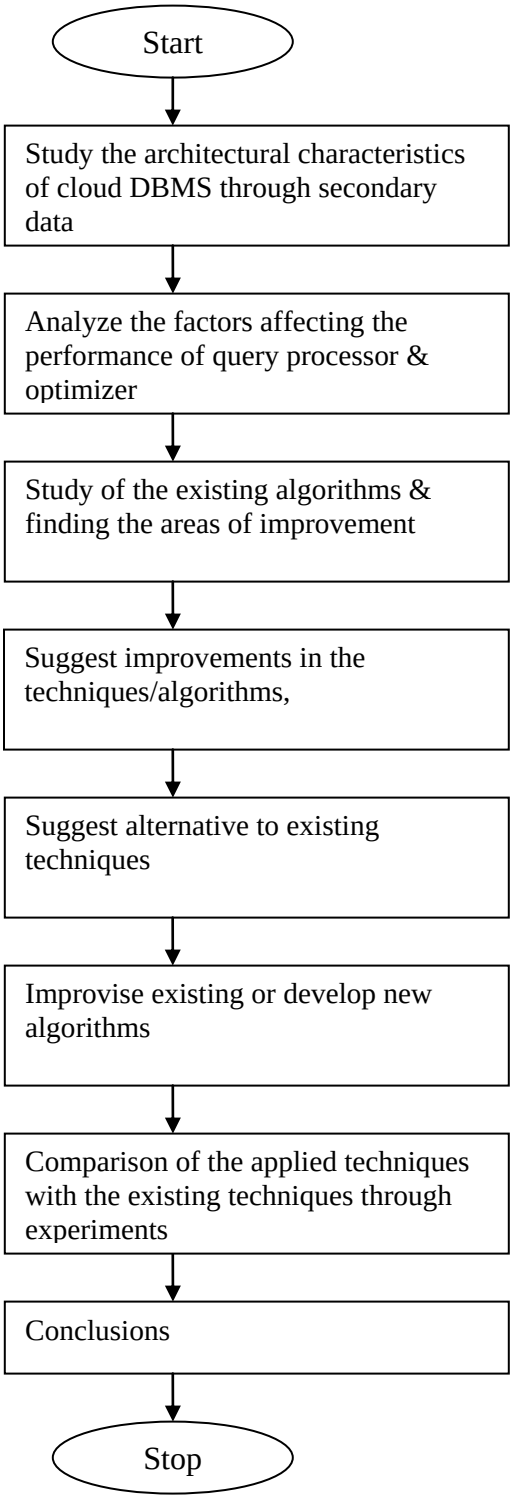
6. Improvise the existing algorithms or design new algorithm(s) for improved efficiency.

   Once the viable alternatives are identified, the refinement and/or development of the algorithm(s) would be taken up.

7. Through experiments on cloud databases, we will compare the performance of the applied technique(s) with the existing techniques.

8. On the basis of the results obtained, conclusion will be drawn.

Flowchart


Start
Study the architectural characteristics of cloud DBMS through secondary data
Analyze the factors affecting the performance of query processor & optimizer
Study of the existing algorithms & finding the areas of improvement
Suggest improvements in the techniques/algorithms,
Suggest alternative to existing techniques
Improvise existing or develop new algorithms
Comparison of the applied techniques with the existing techniques through experiments
Conclusions
Stop

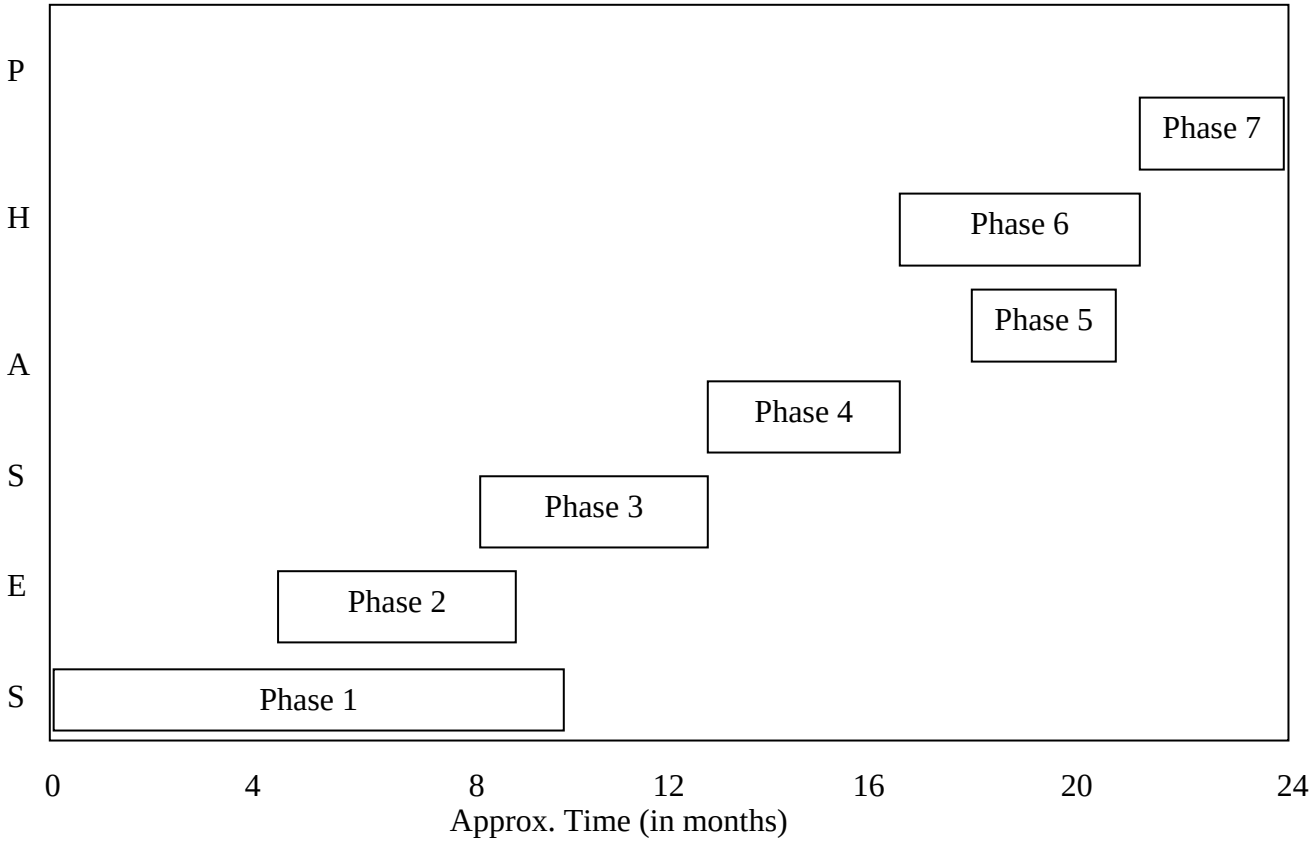
6. Plan of Work

The work plan indicates the projected timing of the next milestone in the research. We have divided our research work into different phases which are as follows:

.

Phase Number	Phase Description	Approximate time duration (in months)
1.	Study the architectural characteristics of cloud databases using secondary sources	8-10
2.	Analyze the factors affecting the performance of query processor in cloud databases	3-4
3.	Analysis of the performance of cloud databases on the basis of their query processing and optimization techniques and study of the existing algorithms of query processor and optimizer.	4-5
4.	Suggest necessary improvement in the existing techniques of processing and optimization	3-4
5.	Suggest a viable alternative to existing techniques for the processing and optimization of queries.	2-3
6.	Design or enhance the algorithm(s), comparisons	4-5
7.	Report generation	2-4

The timeline chart for the above phased plan is:



7. Tools and techniques

We will choose appropriate tools available in the market according to our requirement. Since cloud computing is an emerging field, lot of research is going on to develop the efficient tools and new technologies.

To carry out the proposed research the tools and technologies that may be used include:

1. TPC standards: these could be used for evaluating the performance of database workloads. Some of the current standards are TPC-C, TPC-H, TPC-E.
2. Cloud computing open - source DBMS: LucidDB, MonetDB etc.
3. Graphical tools: SmartDraw, MS-Excel etc.
4. Statistical tools: SPSS
5. Simulator: CloudSim etc.
6. Database tools: DBvisualizer, AquaDataStudio etc.

8. References

[1] Abadi, D. J. (2009), Data Management in the Cloud: Limitations and Opportunities**.** *IEEE **Data** Engineering Bulletin,* **Volume 32**, Number 1, March 2009, pp. 3-12

[2] Abadi D. J., S.R. Madden and M.C. Ferreira (2006),Integrating Compression and Execution in Column-Oriented Database Systems. *Proceedings of the 2006 ACM SIGMOD International conference on Management of data*, ISBN:1-59593-434-0 pp. 671 – 682

[3] Abadi D. J., S.R. Madden and N. Hachem (2008), Column-stores vs. row-stores: how different are they really? *Proceedings of ACM SIGMOD International conference on Management of data*, ISBN:978-1-60558-102-6, pp. 967-980

[4] Abadi, D. J. (2008), Query Execution in Column- Oriented Database Systems. Massachusetts Institute of Technology, U.S.A.

[5] Anh (2009), Query Processing and Optimization. http://cnx.org/content/m28213/latest/

[6] Bobrowski, S. (2011), The Future of Databases is in the Clouds. http://wiki.database.com/page/The_Future_of_Databases_is_in_the_Clouds

[7] Chang, F., J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber (2006), Bigtable: A Distributed Storage System for Structured Data. *Proceedings of the 7th symposium on Operating systems design and implementation*, ISBN:1-931971-47-1, pp. 205 – 218.

[8] Cloud Computing vhttp://www.n-axis.in/practices-cloudcomputing.php

[9] Connolly T. and C. Begg (1998), Database Systems A Practical Approach to Design, Implementation and Management, Pearson Education Ltd., New Delhi, India.

[10] Dean, J. and S. Ghemawat (2004) Mapreduce: Simplified data processing on large clusters. *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation ACM* pp. 137–150

[11] DeCandia, G., D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels(2007) Dynamo: Amazon's highly available key-value store, *In SOSP*, pp. 205–220.

[12] Garret (2010), What is a Column Oriented Database? http://www.columnorienteddatabase.com/

[13] Gounaris, A. (2009), A Vision for Next Generation Query Processors and an Associated Research Agenda. *2nd International Conference on Data Management in Grid and Peer-to-Peer Systems,* pp. 1-11.

[14] Jain, V. (1997), Information Technology, BPB Publications, New Delhi.

[15] Hurwitz, J., R. Bloor, M. Kaufman and F. Halpur (2010), Cloud Computing for Dummies, Wiley Publishing Inc., Indianapolis, Indiana.

[16] Kossmann, D. and K. Stocker (2000)**,** Iterative dynamic programming: a new class of query optimization algorithms. *ACM Transactions on Database Systems (TODS)* **Volume 25, Issue 1**.

[17] Kalnis, P. and D. Papadias (2003) Multi-query optimization for on-line analytical processing*. Information Systems,* **Volume 28, Issue 5,** pp. 457-473

[18] Manegold, S., A.J. Pellenkoft and M.L.Kersten, (2000), A Multi-Query Optimizer for Monet 2000. *Springer-Verlag, 2000 (repository id: 11170).*

[19] Marks, E.A. and B. Lozano (2010), Executive Guide to Cloud Computing, John Wiley & Sons, Inc., Hoboken, New Jersey.

[20] Ramakrishnan, R. and J. Gehrke (2000), Database Management Systems, McGraw-Hill International Editions, Singapore.

[21] Stonebraker, M., D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. R. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik (2005) C-Store: A Column-Oriented DBMS. *In VLDB* Trondheim, Norway, pp. 553–564.

[22] Wu, S., D. Beng and Kun (2010), Efficient B-tree based indexing for cloud data processing., **Volume 3, Issue 1-2**.

[23] Wu, S. and K.L. Wu (2009), An indexing framework for efficient retrieval on the cloud. *IEEE Data Engineering Bulletin*, 32(1):pp. 77–84.

[24] Wiggins, A. (2009), SQL Databases Don't Scale. http://adam.heroku.com/past/2009/7/6/sql_databases_dont_scale/